CS783 Assignment-2

Amrit Singhal (150092)
Department of Computer Science
IIT Kanpur
Kanpur, India
amrits@iitk.ac.in

Aarsh Prakash Agarwal (150004)
Department of Electrical Engineering
IIT Kanpur
Kanpur, India
aarshp@iitk.ac.in

1 Problem Statement

The problem requires us to deploy multi class classification at two levels: coarse and fine. At coarse level, we have 5 broad classes such as aeroplane, bird etc in which the images have to be classified. Each broad class is further divided into several fine classes. The plan to be followed for the procedure, as mentioned in the question, is to have a feature extraction module to extract appropriate features from the images. These features are then fed to a bunch of fully connected layers to get the coarse level classification. The extracted features along with this coarse level classification is then used to obtain the finer level classification through a separate fine-grained classification module.
Multiple approaches were tried by us for each of the three segments of the task, all of which are described below.

2 Feature Extraction

Extracting proper features is a very essential step in image classification. The results of the classification procedures vary greatly with the feature vectors used for classification. We have used the following different types of features vector representation for the images.

1. **Top-k SIFT Descriptors** We detect the top k keypoints from an image and get their SIFT descriptors. All these descriptors are then concatenated together into a single feature vector for the image. The value of k used was 32. The length of the feature vector was $32 \times 128 = 4096$.
2. **Top-k KAZE Descriptors** We also tried using the KAZE detection and description techniques, as described by Alcantarilla *et al.* [2012]. The procedure for getting the feature vector from the descriptors was same as the previous case. Each KAZE descriptor is of size 64, and so the feature vector for th4 image was of size $32 \times 64 = 2048$.
3. **Top-k ORB Descriptors** We also tried using the ORB descriptors, as proposed by Rublee *et al.* [2011], which are a more efficient alternative to SIFT descriptors. Each ORB descriptor is 32 sized, and therefore each image has a feature vector of size $32 \times 32 = 1024$.
4. **SIFT based image histograms** We also implemented the extraction of SIFT based image histogram features, that involved extraction and clustering of all SIFT descriptors, and bagging them into bins get get a feature vector. However, the memory requirement for this method was more than our available resources, and thus this method had to be discontinued.
5. **VGG image features** The classification segment of the VGG architecture, as described by Simonyan and Zisserman [2014], that consists of 3 fully connected layers and a softmax layer present at the last, was discarded, and the weights pre-trained on the imagenet dataset were used for the previous layers. The output of this architecture, having shape $7 \times 7 \times 512$, was flattened and used as a feature vector for the image. The feature vector thus obtained had a very large size of 25088. The architecture of the VGG model is shown in Figure 1.

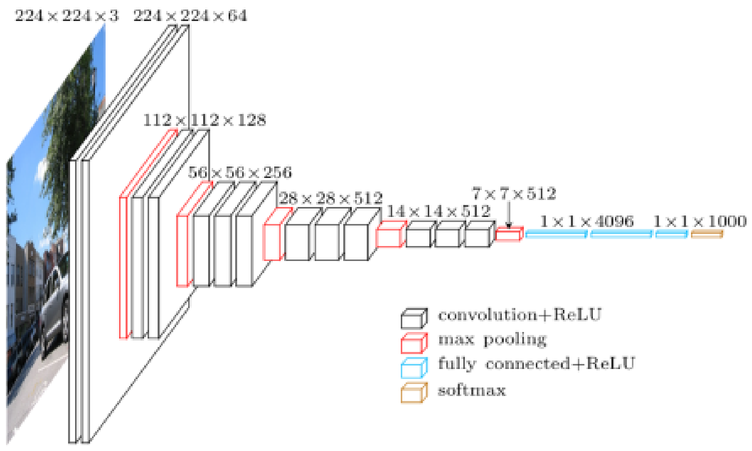

Figure 1: VGG16 architecture

6. **Resnet18 image features** We used the pre-trained weights for the resnet18 model, as proposed by He *et al.* [2015], trained on the imageNet dataset. The last fully connected layer of the resnet18 model that performs the classification was not included, and the output of the previous layer is used as a feature vector representation of the image. The feature vector has size 512. The architecture of ResNet18 is shown in Figure 2.

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 2: ResNet18 architecture

7. **SqueezeNet based features** These were obtained by using the pre-trained weights of the SqueezeNet model, as proposed by Iandola *et al.* [2016], trained on the ImageNet dataset. The architecture consists of a series of convolution and pooling layers, followed by a sequential layer of classification, having a convolution and a pooling layer as well. The architecture of the SqueezeNet model is shown in Figure 3. The classification sequential layer of the SqueezeNet model was discarded (i.e. layer conv10 and following layers), and the output of the previous layer (i.e fire9), having size $13 \times 13 \times 512$, was average pooled using a kernel of size 13, to effectively get a vector of size 512. This is used as the feature vector for the image.

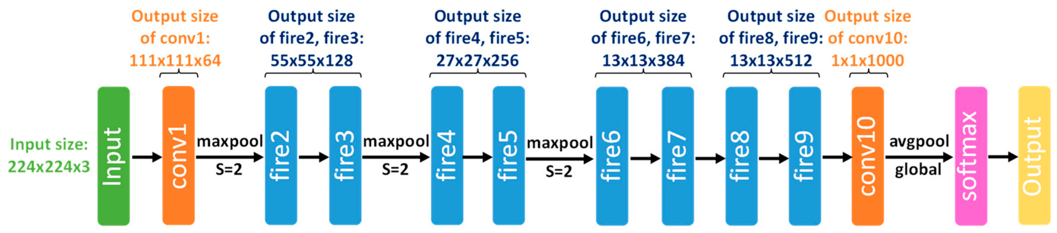
Output size of conv1: 111x111x64
Output size of fire2, fire3: 55x55x128
Output size of fire4, fire5: 27x27x256
Output size of fire6, fire7: 13x13x384
Output size of fire8, fire9: 13x13x512
Output size of conv10: 1x1x1000
Input size: 224x224x3
Input
conv1
maxpool S=2
fire2
fire3
maxpool S=2
fire4
fire5
maxpool S=2
fire6
fire7
fire8
fire9
conv10
avgpool global
softmax
Output

Figure 3: SqueezeNet Architecture

2.1 Parameter Count Analysis

The allowed limit of the number of parameter was two times the number of parameters in the ResNet18 model. A list of number of parameters in all the used models is given below.

Model	Number of parameters
ResNet18	11,689,512
VGG	14,714,688
SqueezeNet	1,248,424

Therefore, the limit on the total number of parametes for us is approximately 11.7m$\times$ 2 = 23.4m

3 Coarse Grained Classification

For the coarse grained classification section, the problem statement specifically demanded the use of one or more fully connected layers over the extracted features. We tried multi-layered perceptron models with varying number of hidden layers for each of the extracted features. The given training was split into training and test sets, in a ratio of $4 : 1$, and accuracy of the model trained on the training segment of the data was evaluated on the test segment. The results for each of them are summarised below.

Features used	**Num of hidden layers**	**Num of nodes in hidden layers**	**Accuracy**	**Number of parameters**
SIFT-top-32	3	[2048, 512, 50]	40.16%	9,463,034
ORB-top-32	3	[2048, 512, 50]	40.04%	3,171,578
KAZE-top-32	3	[2048, 512, 50]	43.02%	5,268,730
KAZE-top-32	3	[3072, 1024, 50]	43.57%	9,488,634
VGG	2	[2048, 50]	99.54%	51,482,874
VGG	1	[200]	98.64%	5,018,600
ResNet18	1	[256]	**99.59%**	132,352
SqueezeNet	1	[256]	98.14%	132,352

Clearly, the CNN based feature extraction methods were able to outperform the standard feature extraction methods by far. Therefore, we proceeded with using only these features for the fine classification module. [1]

[1]The use of VGG features, although giving promising results, could not be continued on the fine classification task due to an increased number of parameters than the allowed limit. A description of the parameter counts required to extract the features is given in Section 2.1.

4 Fine grained classification

For fine grained classification, multiple different techniques were experimented with, some which included the results from the coarse grained classification model explicitly, and others which did not. We now provide a description of each of the methods that was experimented by us.

1. **MLP classifier:** The first method used for the task was similar to the coarse grained classification task, where the extracted features were used to directly train a MLP classifier, having 1 fully connected hidden layer of 256 nodes, on all the 36 sub-classes (rather than the coarse classes). This was trained using different types of features, and the results are as shown below.

Features used	Training iterations	Accuracy
ResNet18	200	82.06%
ResNet18	500	80.20%
SqueezeNet	200	80.41%

   Since, all of the features used had a size of 512, the number of parameters for this classifier was $512 \times 256 + 256 \times 36 = 140,288$.

2. **Using concatenated coarse classification results:** The second method used was to incorporate the coarse class prediction of the coarse grained classifier into the feature vector as a one-hot vector of size 5. This vector was concatenated to the extracted feature vector of the image, to obtain a final feature vector representation of the image, on which the MLP classifier was trained.

Features used	Accuracy
ResNet18	83.29%
SqueezeNet	81.00%

   The number of parameters for this classifier was $(512 + 5) \times 256 + 256 \times 36 = 141,568$. As we could see, incorporating the results of the coarse classification in this manner did not lead to a significant increase in overall accuracy.

3. **Separately trained classifiers for all classes:** Next, we shifted from using a single prediction model for all the sub-classes, to using 5 different fine classification models, one per coarse class. Each fine classifier consisted of a single fully connected hidden layer of size 256. It was trained only on images from that class, and was trained to predict the subclass of the image. The output from the coarse classifier is used to decide which classifier to use for predicting the fine class.
   Even with this method, we tried several variations.

   (a) We tried providing the classifiers with extracted features and trained it to predict the correct sub-classes only from among all the 36 sub-classes. (They were only trained on the data from the selected class).

   (b) We tried to provide the coarse class as a concatenated one-hot input as well to the above method.

   (c) Next, we limited the possible outputs of each classifier explicitly to only its possible sub-classes, thus having a different sized output for all the classifiers. This was trained on the extracted features only, as giving the concatenated coarse classification result to this model seemed redundant.

   For these methods, we obtained separated classification accuracies for the different classes, which are listed below.

Classification Method	**Features used**	**Accuracy**					**Num of parameters**
		aircrafts	*birds*	*cars*	*dogs*	*flowers*	
(a)	ResNet18	88.05%	65.11%	58.33%	97.41%	97.5%	701,440
(a)	SqueezeNet	85.07%	41.86%	54.16%	86.20%	95.86%	701,440
(a)	VGG	76.12%	51.16%	19%	87.13%	92.56%	32,158,720
(b)	ResNet18	87.3%	66.32%	52.77%	97.41%	98.5%	707,840
(b)	SqueezeNet	80.59%	44.18%	63.39%	89.65%	96.69%	707,840
(c)	ResNet18	88.81%	72.09%	55.55%	96.55 %	99.17%	664,576
(c)	SqueezeNet	86.24%	53.14%	41.66%	87.93%	96.67%	664,576

4. **Bilinear CNNs:** Finally, we also experimented with the use of Bilinear CNNs, as proposed by Lin *et al.* [2015], using the ResNet18 and SqueezeNet being the two involved CNN architectures. The features from both CNNs were combined by taking their outer product, which resulted in a $512 \times 512 = 262,144$ sized vector. This was then classified using a fully connected layer classifier into one of the 36 final subclasses. This resulted in an accuracy of 82.68%, which is not any better than those obtained previously.
   The classifier consisted of a single hidden layer of 200 nodes, which resulted in the $262,144 \times 200 + 200 \times 36 = 52,436,000$ number of parameters, which is much higher than the number of allowed parameters. Hence, we can't proceed with using this method finally. We tried lowering the number of nodes in the hidden layer to a smaller number, i.e ~ 40, that would allow the parameters to remain in limit, but this model failed to converge to a decent solution.

5 Details of the final model submission

The final model being used has an architecture as shown in Figure 4. We use the ResNet18 features, and perform a coarse classification using a single fully connected hidden layer of 256 nodes. Using the result of this coarse classification, we decide on which fine classifier to use (the one corresponding to the coarse class of the image). The fine classification technique being used is the use of ResNet18 features to predict the fine class of an image from among the possible fine classes inside its coarse class (i.e. fine classification technique (c)).

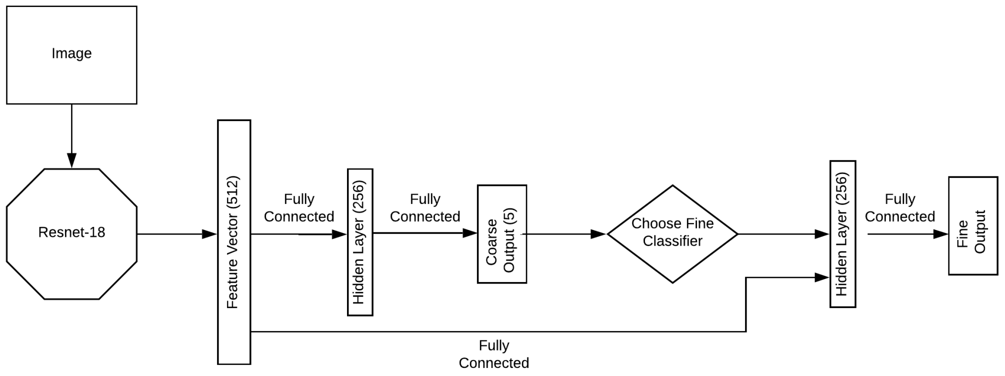

Figure 4: Final Classification network architecture

Thus, the final parameter count of our model is as follows:

Resnet18	$\sim$ 11.7M
Coarse Classification Module	$\sim$ 132k
Fine Classification Module	$\sim$ 664k
Total	$\sim$ 12.5M

The hashes of the final models are as follows:

Model		**Hash Value**
Coarse Classification Module		`baef9cf7b6d9b6cfd41a7a90724d8000`
Fine Classification Module	*aircrafts*	`ef9936f25d6240bffe946f73bc1ba759`
	birds	`7cfb980c85fddced5f787fba94e3d795`
	cars	`b015c41bb3dbf276b0c8de300e3196eb`
	dogs	`7983e48e757473aeb776f6be6be33b49`
	flowers	`a8d0ce3d249d84ca395e93a911934fe5`

References

Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. Kaze features. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 214–227, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

T. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1457, Dec 2015.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. 2011.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.